Trac と mod_wsgi

mod_wsgi は WSGI 互換の Python アプリケーションを Apache 上で直接起動させることができる Apache のモジュールです。 mod_wsgi アダプターは完全に C 言語で書かれており、すばらしいパフォーマンスを提供します。

SilverFrost 目次

trac.wsgi スクリプト

Trac は以下に記すスクリプトによって mod_wsgi のトップで実行されます。これらスクリプトは単なる Python ファイルで、通常は .wsgi という拡張子で保存されます。

基本的なスクリプト

もっともシンプルな形式:

```
import os

os.environ['TRAC_ENV'] = '/usr/local/trac/mysite'
os.environ['PYTHON_EGG_CACHE'] = '/usr/local/trac/mysite/eggs'

import trac.web.main
application = trac.web.main.dispatch_request
```

環境変数 TRAC_ENV は通常通り Trac environment のディレクトリを指定します（複数の Trac environment を含むディレクトリであれば TRAC_ENV_PARENT_DIR を使うこともできます)。 PYTHON_EGG_CACHE は Python eggs を一時的に展開するのに使用するディレクトリを指定します。

複雑なスクリプト

複数の .wsgi ファイルを使用する場合（それぞれのファイルに別個の Trac environment を設定するケースなど）は、 os.environ['TRAC_ENV'] に Trac environment のパスを 設定しない でください。この方法を使うと、別の Trac environment のコンテンツが読み込まれたり、直前に表示した Trac environment のパスが使われてしまうことがあります。

この問題は .wsgi ファイルの内容を下記の通り変更することで回避できます:

```
import os

os.environ['PYTHON_EGG_CACHE'] = '/usr/local/trac/mysite/eggs'

import trac.web.main
def application(environ, start_response):
 environ['trac.env_path'] = '/usr/local/trac/mysite'
 return trac.web.main.dispatch_request(environ, start_response)
```

分かりやすくするために、このファイルの拡張子は .wsgi とすべきです。 Apache にアクセス権を開放できるのであれば、このファイルは自分が所有権を持つディレクトリに置くこともできます。

Trac と egg ファイルをインストールしたパスが通常と異なる場合、それらのパスを以下の要領で wsgi スクリプトの先頭に記述する必要があります:

```
import site
site.addsitedir('/usr/local/trac/lib/python2.4/site-packages')
```

パスはインストールした Trac のライブラリに一致するように変更してください。

推奨される trac.wsgi スクリプト

比較的堅牢で汎用的なファイルを生成するためには trac-admin <env> deploy <dir> コマンドを使用します。コマンドを実行すると必要となるパスが自動的に設定されます（TracInstall#cgi-bin 参照）。

スクリプトのリクエストをマッピングする

.wsgi スクリプトを作成したら、Apache の設定ファイル（例えば httpd.conf）に以下を追記してください。

```
WSGIScriptAlias /trac /usr/local/trac/mysite/apache/mysite.wsgi

<Directory /usr/local/trac/mysite/apache>
```

```
   WSGIApplicationGroup %{GLOBAL}
   Order deny,allow
   Allow from all
</Directory>
```

スクリプトは Trac environment のサブディレクトリにあります。

Trac cgi-bin ディレクトリを生成する のならば、Apache の設定ファイルには以下のように記述してください:

```
WSGIScriptAlias /trac /usr/share/trac/cgi-bin/trac.wsgi

<Directory /usr/share/trac/cgi-bin>
   WSGIApplicationGroup %{GLOBAL}
   Order deny,allow
   Allow from all
</Directory>
```

Apache がスクリプトを起動する為には、スクリプトが含まれるディレクトリまで完全に Apache がアクセスできなければなりません。 `WSGIApplicationGroup` ディレクティブを使用すると、常に `mod_wsgi` が作成した最初の Python インタプリタ内で Trac が起動することが保証されます。これは Trac で使用している Subversion の Python バインディングがサブインタプリタでは動作しないことがあるため必要になります。リクエストがハングし、 Apache がクラッシュしたような結果が返ります。この設定を行った後は Apache を再起動しないと反映されません。

Apache, `mod_wsgi`, Python 本体 (Trac とその依存ライブラリを除く) の設定をテストしたい場合、簡単な `wsgi` アプリケーションを使用するとリクエストが処理されているか確認することができます (以下に示す内容だけを持つ `.wsgi` スクリプトを使用してください):

```
def application(environ, start_response):
       start_response('200 OK',[('Content-type','text/html')])
       return ['<html><body>Hello World!</body></html>']
```

`mod_wsgi` の特定ディレクティブの使用方法についての詳細は、 mod_wsgi's wiki または インストール例 を参照してください。

認証の設定

このセクション内では認証の設定方法をいくつか記述します。

Apache ガイドの 認証、許可、アクセスコントロール も参照してください。

基本認証

Apache で認証を追加する最も簡単な方法は、パスワードファイルを作成することです。 `htpasswd` プログラムを使用してパスワードファイルを作成します:

```
$ htpasswd -c /somewhere/trac.htpasswd admin
New password: <type password>
Re-type new password: <type password again>
Adding password for user admin
```

一番最初のユーザ以外は "-c" オプションは必要ありません:

```
$ htpasswd /somewhere/trac.htpasswd john
New password: <type password>
Re-type new password: <type password again>
Adding password for user john
```

`htpasswd` についての詳細は man を参照してください。

ユーザを作成した後、TracPermissions の記述通りユーザに権限を設定することができます。

Apache の設定ファイルの中にパスワードファイルを記述し、認証を有効にする必要があります:

```
<Location "/trac/login">
 AuthType Basic
 AuthName "Trac"
 AuthUserFile /somewhere/trac.htpasswd
 Require valid-user
</Location>
```

複数のプロジェクトを持っている場合でも、共通のパスワードファイルを使用することができます:

```
<LocationMatch "/trac/[^/]+/login">
 AuthType Basic
 AuthName "Trac"
 AuthUserFile /somewhere/trac.htpasswd
 Require valid-user
</LocationMatch>
```

Note: 'login' という名のファイルやディレクトリが必要なわけではありません。
mod_auth_basic も参照してください。

ダイジェスト認証

セキュリティ強化のために、SSL を有効にするか、少なくとも "基本認証" の代わりに "ダイジェスト認証" を使用することを推奨します。

以下のように、`htpasswd` の代わりに `htdigest` コマンドを使用して `.htpasswd` ファイルを作成してください:

```
# htdigest -c /somewhere/trac.htpasswd trac admin
```

上記の "trac" パラメータは "realm" です。Apache の設定ファイルの AuthName ディレクティブで再度指定してください:

```
<Location "/trac/login">

   AuthType Digest
   AuthName "trac"
   AuthDigestDomain /trac
   AuthUserFile /somewhere/trac.htpasswd
   Require valid-user
</Location>
```

複数の environment を持っている場合、上記に記述した方法で、同じ `LocationMatch` を使用することができます。

必ず mod_auth_digest をアクティブにしてください。 Debian 4.0r1 (etch) の例:

```
    LoadModule auth_digest_module /usr/lib/apache2/modules/mod_auth_digest.so
```

mod_auth_digest も参照してください。

LDAP 認証

Apache の mod_ldap 認証の設定ファイルは少し扱いにくいです。(httpd 2.2.x OpenLDAP: slapd 2.3.19)

1. Apache の httpd.conf に以下のモジュールをロードする必要があります

```
LoadModule ldap_module modules/mod_ldap.so
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
```

1. httpd.conf は以下のようになります:

```
<Location /trac/>
 # (if you're using it, mod_python specific settings go here)
 Order deny,allow
```

```
 Deny from all
 Allow from 192.168.11.0/24
 AuthType Basic
 AuthName "Trac"
 AuthBasicProvider "ldap"
 AuthLDAPURL "ldap://127.0.0.1/dc=example,dc=co,dc=ke?uid?sub?(objectClass=inetOrgPerson)"
 authzldapauthoritative Off
 Require valid-user
</Location>
```

1. Microsoft Active Directory の LDAP インターフェースを使用することもできます:

以下を LDAP URL に使用します:

```
    AuthLDAPURL "ldap://directory.example.com:3268/DC=example,DC=com?sAMAccountName?sub?(objectClass=user)"
```

認証情報をチェックするために Apache にアカウントを提供する必要があります。
このパスワードは設定ファイル内のプレーンテキストにリストアップされるので、 このタスク専用のアカウントを使用すべきです:

```
    AuthLDAPBindDN ldap-auth-user@example.com
   AuthLDAPBindPassword "password"
```

セクション全体はこのようになります:

```
<Location /trac/>
 # (if you're using it, mod_python specific settings go here)
 Order deny,allow
 Deny from all
 Allow from 192.168.11.0/24
 AuthType Basic
 AuthName "Trac"
 AuthBasicProvider "ldap"
 AuthLDAPURL "ldap://adserver.company.com:3268/DC=company,DC=com?sAMAccountName?sub?(objectClass=user)"
 AuthLDAPBindDN       ldap-auth-user@company.com
 AuthLDAPBindPassword "the_password"
 authzldapauthoritative Off
 # require valid-user
 require ldap-group CN=Trac Users,CN=Users,DC=company,DC=com
</Location>
```

Note 1: このケースでは LDAP 検索で複数の OU をまとめて取得するために、 AD のグローバルカタログサーバ (Global Catalog Server) に接続しています (ポート番号が通常 LDAP で使用される 389 ではなく 3268 であることに注意してください) 。 GCS は基本的に "平らな" ツリーであり、ユーザが、どの OU に属するか不明な場合でも検索することができます。

Note 2: 有効なログインを持っているかの代わりに、 LDAP グループに所属しているかを要求することができます:

```
    Require ldap-group CN=Trac Users,CN=Users,DC=example,DC=com
```

関連ページ:

- mod_authnz_ldap, mod_authnz_ldap に関するドキュメンテーション
- mod_ldap, mod_ldap に関するドキュメンテーション。コネクションプールや共有のキャッシュを提供します
- TracHacks:LdapPlugin, LDAP の TracPermissions を格納するプラグイン

SSPI 認証

Windows 上で Apache を使用しているのであれば、シングルサインオン機能を提供する mod_auth_sspi を使用することができます。 SourceForge の mod-auth-sspi プロジェクト からモジュールをダウンロードし、 バーチャルホストに以下を追記してください。

```
    <Location /trac/login>
       AuthType SSPI
       AuthName "Trac Login"
       SSPIAuth On
       SSPIAuthoritative On
       SSPIDomain MyLocalDomain
       SSPIOfferBasic On
       SSPIOmitDomain Off
       SSPIBasicPreferred On
       Require valid-user
   </Location>
```

SSPI 認証を使用すると、Trac のユーザ名が DOMAIN\username という形式になるので、
パーミッションなどを追加しなおす必要があります。ユーザ名にドメインを入れたくない場合は、 SSPIOmitDomain On
と代わりに設定してください。

SSPI 認証に関する共通の問題: 本家チケット 1055, 本家チケット 1168, 本家チケット 3338 。

TracOnWindows/Advanced も参照してください。

AccountManagerPlugin のログインフォームを使用した Apache 認証 ===
#UsingApacheauthenticationwiththeAccountManagerplugin'sLoginform

まず、AccountManagerPlugin の Login モジュール と HttpAuthStore authentication モジュール の基本的な仕様を参照してください。

Note: acct_mgr-0.4 より前のバージョンの AccountManager を使用していると、WSGI で HttpAuthStore
を動かすことは難しいです。アップグレードを推奨します。

単一のプロジェクトに acct_mgr-0.4 を使用した例:

```
[components]
; be sure to enable the component
acct_mgr.http.HttpAuthStore = enabled

[account-manager]
; configure the plugin to use a page that is secured with http authentication
authentication_url = /authFile
password_store = HttpAuthStore
```

Apache の設定ファイルはこのようになります:

```
<Location /authFile>
  …HTTP authentication configuration…
  Require valid-user
</Location>
```

authFile が存在する必要はありません。前述の HttpAuthStore へのリンク先から、複数の Trac
プロジェクトをサーバー上でホスティングした場合を説明する箇所などを参照してください。

例: バーチャルホストのルートが Trac である Apache/mod_wsgi 基本認証 ===
#Example:Apache/mod_wsgiwithBasicAuthentication,Trac beingattherootofavirtual host

上記の mod_wsgi のドキュメントには、 Apache の設定例 a) Trac のインスタンスをバーチャルホストでサブドメインを作成して動かす例と b) Trac
の認証として、 Apache の基本認証を設定する例が記載されています。

例えば、 trac を http://trac.my-proj.my-site.org としてホストし、/home/trac-for-my-proj フォルダから起動する場合で、 the-env
を作成するために、 trac-admin the-env initenv コマンドを使用し、 the-deploy フォルダを作成するために、 trac-admin the-env
deploy the-deploy コマンドを使用した場合です:

htpasswd ファイルを作成します:

```
cd /home/trac-for-my-proj/the-env
htpasswd -c htpasswd firstuser
### and add more users to it as needed:
htpasswd htpasswd seconduser
```

(セキュリティのため、このファイルはドキュメントルートにおきます)

以下の設定を含んだファイルを作成します。例 /etc/apache2/sites-enabled/trac.my-proj.my-site.org.conf (ubuntu):

```
<Directory /home/trac-for-my-proj/the-deploy/cgi-bin/trac.wsgi>
 WSGIApplicationGroup %{GLOBAL}
 Order deny,allow
 Allow from all
</Directory>

<VirtualHost *:80>
 ServerName trac.my-proj.my-site.org
 DocumentRoot /home/trac-for-my-proj/the-env/htdocs/
 WSGIScriptAlias / /home/trac-for-my-proj/the-deploy/cgi-bin/trac.wsgi
 <Location '/'>
   AuthType Basic
   AuthName "Trac"
   AuthUserFile /home/trac-for-my-proj/the-env/htpasswd
   Require valid-user
 </Location>
</VirtualHost>
```

Note: サブドメインが適切に動くようにするには、 /etc/hosts ファイルの変更や、ホストサーバの DNS の A レコードにサブドメインを追加する必要があります。

トラブルシューティング

最新バージョンを使う

mod_wsgi のバージョンは 1.6 か 2.4 かそれより新しいものを使用してください。2.4 以前のバージョン 2.X のブランチは WSGI ファイルラッパー拡張機能を使用する Apache の設定にいくつか問題があります。この拡張機能は Trac 内でスタイルシートのような添付ファイルや静的メディアファイルを提供するのに使用します。この問題の影響を受けると、添付ファイルは何もないように見えてし HTML ページのフォーマットが機能していないように見えてしまったりします。他に 2 進数の添付ファイルが省略されてしまうといった問題も頻繁に起こります。mod_wsgi に関するチケット #100 と #132 を参照してください。

Note: mod_wsgi 2.5 と Python 2.6.1 を使用していると、システム上 (Apache 2.2.11 と Trac 0.11.2.1) で Internal Server Error が発生しました。Python 2.6.2 にアップグレードすることで(こちら で勧められています) 私の場合は解決しました。
-- Graham Shanks

もし 'mod_wsgi' を Windows の組み込みモードでの使用や Linux 上で MPM worker と併用することを予定している場合, バージョン 0.3.4 以上が必要です。 (詳細については、 #10675 を参照してください)

SSPI および 'Require Group' 使用時に Trac を動かす方法

Trac を Win32 上の Apache で起動し、 SSPI 設定して 'Require group' オプションを構成している場合、 'SSPIOmitDomain' オプションはおそらく動作しません。 Trac にユーザ名が認識されない場合は、 'user' が 'DOMAIN / user' のように見えている可能性があります。

このような場合、以下のように WSGI スクリプトを修正すると解決すると思います:

```
import os
import trac.web.main

os.environ['TRAC_ENV'] = '/usr/local/trac/mysite'
os.environ['PYTHON_EGG_CACHE'] = '/usr/local/trac/mysite/eggs'
```

```
def application(environ, start_response):
   if "\\" in environ['REMOTE_USER']:
       environ['REMOTE_USER'] = environ['REMOTE_USER'].split("\\", 1)[1]
   return trac.web.main.dispatch_request(environ, start_response)
```

Trac と PostgreSQL

mod_wsgi アダプタを使用し、 Trac のインスタンスを複数ホストしている場合に、 PostgreSQL (もしかすると MySQL も?)をデータベースバックエンドとして使用していると、大量のデータベース接続が生成され、PostgreSQL のプロセスも大量に発生してしまうかもしれません。

荒々しい方法ですが解決策として、Trac が持つコネクションプールを無効化する方法があります。これは `trac.db.postgres_backend` の `PostgreSQLConnection` クラスに `poolable = False` と設定することで適用できます。

この方法を適用するために、 Trac のソースを変更する必要はありません。以下に示す行を `trac.wsgi` に追加してください:

```
import trac.db.postgres_backend
trac.db.postgres_backend.PostgreSQLConnection.poolable = False
```

または

```
import trac.db.mysql_backend
trac.db.mysql_backend.MySQLConnection.poolable = False
```

この設定で Trac ページを生成した後にコネクションを捨てるようになり、データベースへの接続数は最小に保たれます。

この方法は推奨されたものではありません。 mod_wsgi の IntegrationWithTrac ページの最後も参照してください。

その他の資料

mod_python のトラブルシューティング セクションも参照してください。 Apache に関連する問題の多くは似通っていて、多くの場合 mod_wsgi を使用する アプリケーション側の問題 です。 wsgi ページにも Integration With Trac ドキュメントがあります。

See also: TracGuide, TracInstall, FastCGI, ModPython, TracNginxRecipe